

AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

- 1 1. (Currently amended) A method to indicate to a dynamic compiler of a
2 multitasking virtual machine when the dynamic compiler can skip generating
3 native code for a class initialization barrier when compiling a program method,
4 wherein a runtime representation of classes by the multitasking virtual machine
5 includes a shared runtime data structure that is shared by multiple tasks, and
6 wherein a native code produced by the dynamic compiler extends the shared
7 runtime data structure representing classes and can be executed serially or
8 concurrently by multiple tasks of the multitasking virtual machine, the method
9 comprising:
10 augmenting the shared runtime data structure with an initializer field,
11 wherein the shared runtime data structure represents a shared part of a class, and
12 wherein the initializer field is a variable; and
13 using the value of the initializer field of the class to determine whether a
14 platform-independent instruction of the program method may trigger an
15 initialization of the class.
- 1 2. (Original) The method of claim 1, further comprising initializing a
2 bootstrap class, wherein the bootstrap class is initialized during startup of a task of
3 the multitasking virtual machine, and before any concurrency, due to creation of
4 multiple threads of control within the task, takes place.

1 3. (Original) The method of claim 2, further comprising assigning a value
2 of an initializer of the class when the class is fully initialized, wherein the value
3 includes an indication that one of:
4 the class is not the bootstrap class,
5 the class is the bootstrap class and the value of the initializer identifies a
6 holder of the class initialization barrier that triggered the initialization of the class,
7 and
8 the class is the bootstrap class and the value of the initializer further
9 indicates that the class initialization was not triggered by a class initialization
10 barrier.

1 4. (Original) The method of claim 3, further comprising:
2 setting a binary variable to zero upon starting the multitasking virtual
3 machine; and
4 setting the binary variable to one when all bootstrap classes have been
5 initialized by a first task executed by the multitasking virtual machine;
6 whereby the binary variable indicates to the multitasking virtual machine
7 whether all bootstrap classes have been initialized.

1 5. (Original) The method of claim 4, further comprising:
2 upon initiating the initialization of the class from a class initialization
3 barrier, noting the holder of the class initialization barrier; and
4 once the class is fully initialized, assigning the holder to the initializer
5 field only if the binary variable is zero.

1 6. (Original) The method of claim 3, further comprising upon setting a
2 non-bootstrap class to a fully initialized state for a task, assigning the initializer

3 field of the class to a constant value, wherein the constant value is distinguishable
4 from all other possible values for the initializer field.

1 7. (Original) The method of claim 6, wherein the constant value is a NULL
2 pointer.

1 8. (Original) The method of claim 3, wherein a pointer to a runtime data
2 structure representing the shared part of the class is assigned to the initializer field
3 of the class to indicate that the class is the bootstrap class whose initialization is
4 not triggered by a class initialization barrier.

1 9. (Original) The method of claim 3, further comprising:
2 if the class does not have an initialization sequence,
3 setting the class to a fully initialized state upon the class
4 being loading by the task, and
5 assigning the initializer field of the class to a pointer to a
6 runtime data structure representing the shared part of the class.

1 10. (Original) The method of claim 3, further comprising instructing the
2 dynamic compiler not to generate native code for the class initialization barrier of
3 the program method being compiled if the class targeted by the class initialization
4 barrier is equal to the class that defines the program method being compiled.

1 11. (Original) The method of claim 10, further comprising instructing the
2 dynamic compiler not to generate native code for the class initialization barrier of
3 the program method being compiled if the class targeted by the class initialization
4 barrier is a superclass of the class that defines the program method being
5 compiled.

1 12. (Original) The method of claim 11, wherein the value of the holder of
2 the class initialization barrier is a pointer to a runtime data structure representing
3 the shared part of the class that defines the program method that holds the class
4 initialization barrier.

1 13. (Original) The method of claim 12, further comprising instructing the
2 dynamic compiler not to generate native code for the class initialization barrier of
3 the program method being compiled if the value of the initializer field of the class
4 targeted by the class initialization barrier is:

5 different from the value that indicates that the class is not bootstrap class,
6 and

7 different from the pointer to the runtime data structure representing the
8 shared part of the class that defines the program method being compiled.

1 14. (Original) The method of claim 11, wherein the value of the holder of
2 the class initialization barrier is a pointer to the shared runtime data structure
3 representing the program method that holds the class initialization barrier.

1 15. (Original) The method of claim 14, further comprising instructing the
2 dynamic compiler not to generate native code for the class initialization barrier of
3 the program method being compiled if the value of the initializer field of the class
4 targeted by the class initialization barrier is:

5 different from the value that indicate that the class is not the bootstrap
6 class, and

7 different from the pointer to the shared runtime data structure representing
8 the program method being compiled.

1 16. (Currently amended) A computer-readable storage medium storing
2 instructions that when executed by a computer cause the computer to perform a
3 method to indicate to a dynamic compiler of a multitasking virtual machine when
4 the dynamic compiler can skip generating native code for a class initialization
5 barrier when compiling a program method, wherein a runtime representation of
6 classes by the multitasking virtual machine includes a shared runtime data
7 structure that is shared by multiple tasks, and wherein a native code produced by
8 the dynamic compiler extends the shared runtime data structure representing
9 classes and can be executed serially or concurrently by multiple tasks of the
10 multitasking virtual machine, the method comprising:

11 augmenting the shared runtime data structure with an initializer field,
12 wherein the shared runtime data structure represents a shared part of a class, and
13 wherein the initializer field is a variable; and
14 using the value of the initializer field of the class to determine whether a
15 platform-independent instruction of the program method may trigger an
16 initialization of the class.

1 17. (Original) The computer-readable storage medium of claim 16, the
2 method further comprising initializing a bootstrap class, wherein the bootstrap
3 class is initialized during startup of a task of the multitasking virtual machine, and
4 before any concurrency, due to creation of multiple threads of control within the
5 task, takes place.

1 18. (Original) The computer-readable storage medium of claim 17, the
2 method further comprising assigning a value of an initializer of the class when the
3 class is fully initialized, wherein the value includes an indication that one of:
4 the class is not the bootstrap class,

5 the class is the bootstrap class and the value of the initializer identifies a
6 holder of the class initialization barrier that triggered the initialization of the class,
7 and
8 the class is the bootstrap class and the value of the initializer further
9 indicates that the class initialization was not triggered by a class initialization
10 barrier.

1 19. (Original) The computer-readable storage medium of claim 18, the
2 method further comprising:
3 setting a binary variable to zero upon starting the multitasking virtual
4 machine; and
5 setting the binary variable to one when all bootstrap classes have been
6 initialized by a first task executed by the multitasking virtual machine;
7 whereby the binary variable indicates to the multitasking virtual machine
8 whether all bootstrap classes have been initialized.

1 20. (Original) The computer-readable storage medium of claim 19, the
2 method further comprising:
3 upon initiating the initialization of the class from a class initialization
4 barrier, noting the holder of the class initialization barrier; and
5 once the class is fully initialized, assigning the holder to the initializer
6 field only if the binary variable is zero.

1 21. (Original) The computer-readable storage medium of claim 18, the
2 method further comprising upon setting a non-bootstrap class to a fully initialized
3 state for a task, assigning the initializer field of the class to a constant value,
4 wherein the constant value is distinguishable from all other possible values for the
5 initializer field.

1 22. (Original) The computer-readable storage medium of claim 21,
2 wherein the constant value is a NULL pointer.

1 23. (Original) The computer-readable storage medium of claim 18,
2 wherein a pointer to a runtime data structure representing the shared part of the
3 class is assigned to the initializer field of the class to indicate that the class is the
4 bootstrap class whose initialization is not triggered by a class initialization barrier.

1 24. (Original) The computer-readable storage medium of claim 18, the
2 method further comprising:
3 if the class does not have an initialization sequence,
4 setting the class to a fully initialized state upon the class
5 being loading by the task, and
6 assigning the initializer field of the class to a pointer to a
7 runtime data structure representing the shared part of the class.

1 25. (Original) The computer-readable storage medium of claim 18, the
2 method further comprising instructing the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the class targeted by the class initialization barrier is equal to the class
5 that defines the program method being compiled.

1 26. (Original) The computer-readable storage medium of claim 25, the
2 method further comprising instructing the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the class targeted by the class initialization barrier is a superclass of
5 the class that defines the program method being compiled.

1 27. (Original) The computer-readable storage medium of claim 26,
2 wherein the value of the holder of the class initialization barrier is a pointer to a
3 runtime data structure representing the shared part of the class that defines the
4 program method that holds the class initialization barrier.

1 28. (Original) The computer-readable storage medium of claim 27, the
2 method further comprising instructing the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the value of the initializer field of the class targeted by the class
5 initialization barrier is:
6 different from the value that indicates that the class is not bootstrap class,
7 and
8 different from the pointer to the runtime data structure representing the
9 shared part of the class that defines the program method being compiled.

1 29. (Original) The computer-readable storage medium of claim 26,
2 wherein the value of the holder of the class initialization barrier is a pointer to the
3 shared runtime data structure representing the program method that holds the class
4 initialization barrier.

1 30. (Original) The computer-readable storage medium of claim 29, the
2 method further comprising instructing the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the value of the initializer field of the class targeted by the class
5 initialization barrier is:
6 different from the value that indicate that the class is not the bootstrap
7 class, and

8 different from the pointer to the shared runtime data structure representing
9 the program method being compiled.

1 31. (Currently amended) An apparatus to indicate to a dynamic compiler
2 of a multitasking virtual machine when the dynamic compiler can skip generating
3 native code for a class initialization barrier when compiling a program method,
4 wherein a runtime representation of classes by the multitasking virtual machine
5 includes a shared runtime data structure that is shared by multiple tasks, and
6 wherein a native code produced by the dynamic compiler extends the shared
7 runtime data structure representing classes and can be executed serially or
8 concurrently by multiple tasks of the multitasking virtual machine, comprising:
9 an augmenting mechanism that is configured to augment the shared
10 runtime data structure with an initializer field, wherein the shared runtime data
11 structure represents a shared part of a class, and wherein the initializer field is a
12 variable; and
13 a determining mechanism that is configured to use the value of the
14 initializer field of the class to determine whether a platform-independent
15 instruction of the program method may trigger an initialization of the class.

1 32. (Original) The apparatus of claim 31, further comprising an initializing
2 mechanism that is configured to initialize a bootstrap class, wherein the bootstrap
3 class is initialized during startup of a task of the multitasking virtual machine, and
4 before any concurrency, due to creation of multiple threads of control within the
5 task, takes place.

1 33. (Original) The apparatus of claim 32, further comprising an assigning
2 mechanism that is configured to assign a value of an initializer of the class when
3 the class is fully initialized, wherein the value includes an indication that one of:

4 the class is not the bootstrap class,
5 the class is the bootstrap class and the value of the initializer identifies a
6 holder of the class initialization barrier that triggered the initialization of the class,
7 and
8 the class is the bootstrap class and the value of the initializer further
9 indicates that the class initialization was not triggered by a class initialization
10 barrier.

1 34. (Original) The apparatus of claim 33, further comprising:
2 a setting mechanism that is configured to set a binary variable to zero upon
3 starting the multitasking virtual machine;
4 wherein the setting mechanism is further configured to set the binary
5 variable to one when all bootstrap classes have been initialized by a first task
6 executed by the multitasking virtual machine;
7 whereby the binary variable indicates to the multitasking virtual machine
8 whether all bootstrap classes have been initialized.

1 35. (Original) The apparatus of claim 34, further comprising:
2 an examining mechanism that is configured to note the holder of the class
3 initialization barrier; and
4 wherein the assigning mechanism is further configured to assign the
5 holder to the initializer field only if the binary variable is zero.

1 36. (Original) The apparatus of claim 33, wherein the assigning
2 mechanism is further configured to assign the initializer field of the class to a
3 constant value, wherein the constant value is distinguishable from all other
4 possible values for the initializer field.

1 37. (Original) The apparatus of claim 36, wherein the constant value is a
2 NULL pointer.

1 38. (Original) The apparatus of claim 33, wherein a pointer to a runtime
2 data structure representing the shared part of the class is assigned to the initializer
3 field of the class to indicate that the class is the bootstrap class whose
4 initialization is not triggered by a class initialization barrier.

1 39. (Original) The apparatus of claim 33, further comprising a setting
2 mechanism that is configured to set the class to a fully initialized state upon the
3 class being loading by the task; and
4 wherein the assigning mechanism is further configured to assign the
5 initializer field of the class to a pointer to a runtime data structure representing the
6 shared part of the class.

1 40. (Original) The apparatus of claim 33, further comprising an instructing
2 mechanism that is configured to instruct the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the class targeted by the class initialization barrier is equal to the class
5 that defines the program method being compiled.

1 41. (Original) The apparatus of claim 40, wherein the instructing
2 mechanism is further configured to instruct the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the class targeted by the class initialization barrier is a superclass of
5 the class that defines the program method being compiled.

1 42. (Original) The apparatus of claim 41, wherein the value of the holder
2 of the class initialization barrier is a pointer to a runtime data structure
3 representing the shared part of the class that defines the program method that
4 holds the class initialization barrier.

1 43. (Original) The apparatus of claim 42, wherein the instructing
2 mechanism is further configured to instruct the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the value of the initializer field of the class targeted by the class
5 initialization barrier is:

6 different from the value that indicates that the class is not bootstrap class,
7 and
8 different from the pointer to the runtime data structure representing the
9 shared part of the class that defines the program method being compiled.

1 44. (Original) The apparatus of claim 41, wherein the value of the holder
2 of the class initialization barrier is a pointer to the shared runtime data structure
3 representing the program method that holds the class initialization barrier.

1 45. (Original) The apparatus of claim 44, wherein the instructing
2 mechanism is further configured to instruct the dynamic compiler not to generate
3 native code for the class initialization barrier of the program method being
4 compiled if the value of the initializer field of the class targeted by the class
5 initialization barrier is:

6 different from the value that indicate that the class is not the bootstrap
7 class, and
8 different from the pointer to the shared runtime data structure representing
9 the program method being compiled.